Volume 4, Issue 1, January 2014 ISSN: 2277 128X

International Journal of Advanced Research in Computer Science and Software Engineering

Research Paper

Available online at: www.ijarcsse.com

Virtual Machine Placement in Cloud Using Multi-Objective Ant Colony System

Jeyashree V
Department of CSE
B.S.Abdur RahmanUniversity
Chennai, India.

G.Kamala Nandhini
Department of CSE
Anna University
Chennai, India.

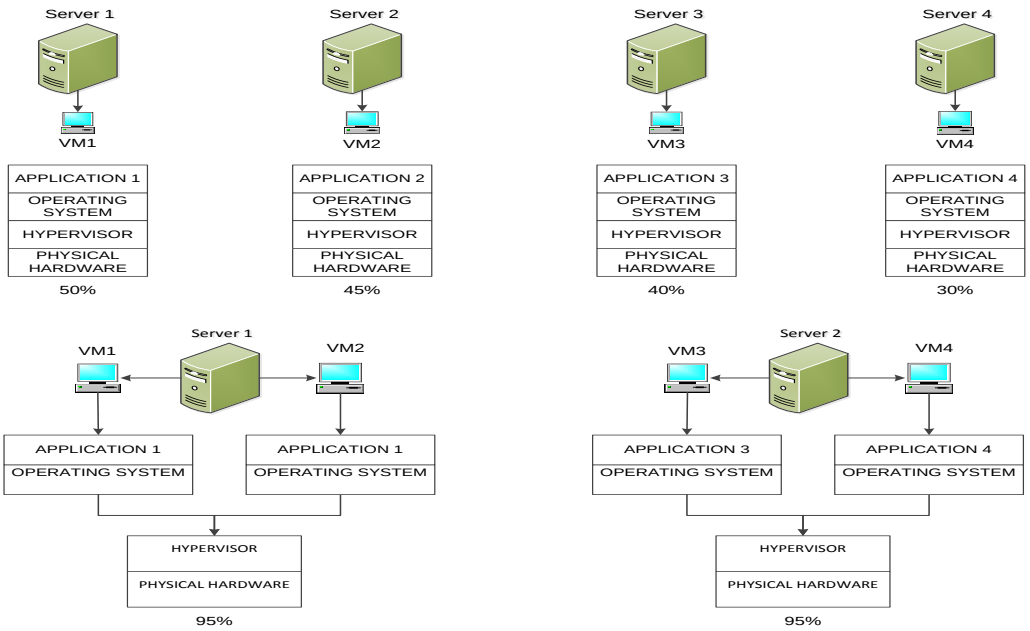
G.Amamanickam
Department of CSE
B.S.Abdur Rahman University
Chennai, India.

Abstract- Virtual machine placement is a process of mapping virtual machines to physical machines. The optimal placement is important for improving power efficiency and resource utilization in a cloud computing environment. In this paper, we propose a multi-objective ant colony system algorithm for the virtual machine placement problem. The goal is to efficiently obtain a set of non-dominated solutions (the Pareto set) that simultaneously minimize total resource wastage and power consumption. The proposed algorithm is tested with some instances from the literature. Its solution performance is compared to that of an existing algorithm. The results show that the proposed algorithm is more efficient and effective than the methods we compared it to.

Keywords: Multi-objective optimization, Ant colony optimization Virtual machine placement, Cloud computing

1. INTRODUCTION

In recent year, cloud computing has become a popular computing paradigm for hosting and delivering services over the Internet [1]. There are three major types of cloud computing: infrastructure as a service (IaaS), platform as a service (PaaS) and software as a service (SaaS). The adoption and deployment of cloud computing platforms have many attractive benefits, such as reliability, quality of service and robustness [2]. To the consumer, the cloud appears to be infinite, and the consumer can purchase as much or as little computing power as they need. From a provider's perspective, the key issue is to maximize profits by minimizing the operational costs. In this regard, power management in cloud data centers is becoming a crucial issue since it dominates the operational costs. Moreover, power consumption in large-scale computer systems like clouds also raises many other serious issues including carbon dioxide and system reliability. The emergence of cloud computing has made a tremendous impact on the information technology (IT) industry over the past few years, where large companies such as Amazon, Google, Salesforce, IBM, Microsoft, and Oracle have begun to establish new data centers for hosting cloud computing applications in various locations around the world to provide redundancy and ensure reliability in case of site failures. There are a number of key technologies that make cloud computing possible. One of the most important is virtualization.



VM Placement in a virtualized environment

Virtualization provides a promising approach through which hardware resources on one or more machines can be divided through partial or complete machine simulation, time-sharing, hardware and software partitioning into multiple execution environments, each of which can act as a complete system. Virtualization enables dynamic sharing of physical resources in cloud computing environments, allowing multiple applications to run in different performance-isolated platforms called virtual machines (VMs) in a single physical server. This technology also enables on-demand or utility computing—a just-in-time resource provisioning model in which computing resources such as CPU, memory, and disk space are made available to applications only as needed and not allocated statically based on the peak workload demand [3]. Through virtualization, a cloud provider can ensure the quality of service (QoS) delivered to the users while achieving a high server utilization and energy efficiency.

Virtual machine placement is a process of mapping virtual machines to physical machines. As virtualization is a core technology of cloud computing, the problem of virtual machine (VM) placement has become a hot topic recently. This VM placement is an important approach for improving power efficiency and resource utilization in cloud infrastructures. Several research works [4,5] addressed the importance of placing VMs appropriately. Vogels [6] quoted the benefit of packing VMs efficiently in server consolidation. The proxy placement [7–9] and object placement/replacement [10,11] for transparent data replication bear some resemblance to the issues we face since they all attempt to exploit the flexibility available in determining proper placement. The following are some of the approaches that have been used to solve the virtual machine placement problem.

2. Background

An example of VM placement in a virtualized cloud environment

For example, let us consider the situation depicted in Fig.1. We have seven servers, each of which has a quad-core processor which is capable of executing four VMs. The system is currently hosting seven virtualized application labeled Application 1 to Application 7. A simple process for VM placement is as follows [25].

(1) For each server, compute application resource requirement using server's resource usage statistics over a period of time (e.g., several weeks).

(2) Choose a target server with compatible virtualization software, comparable CPU types, similar network connectivity, and usage of shared storage.

(3) Place the first virtual machine on the first server in step 2. Place the second virtual machine on the same server if it can satisfy the resource requirements. If not, add a new physical machine and place the VM on this new machine. Continue this step until each of the VMs has been placed on a physical machine, adding a new physical machine when required.

(4) The set of resulting hosts at the end of step 3 comprises the consolidated server farm. Finally the number of servers required in the cluster is reduced from 7 down to 3.

2.2. Ant colony optimization

Ant Colony Optimization (ACO) is a metaheuristic inspired by the observation of real ant colonies and based upon their collective foraging behavior [26]. Ants are social insects and live in colonies. Their behavior is governed by the goal of colony survival. When searching for food, ants frequently travel between their nest and food sources. At the beginning, ants explore the area surrounding their nest in a random manner. While moving, ants deposit special substances called pheromones along their paths. Ants can smell pheromones. When choosing their way, they tend to choose, in probability, paths marked by strong pheromone concentrations. As soon as an ant finds a food source, it evaluates the quantity and the quality of the food and carries some of it back to the nest. During the return trip, the quantity of pheromones that an ant leaves on the ground may depend on the quantity and quality of the food. The pheromone trails will guide other ants to the food source. The indirect communication between the ants via pheromone trails enables them to find the shortest paths between their nest and food sources. Ant colony optimization has been successfully applied to solve numerous optimization problems such as the traveling salesman problem [27], the flow shop scheduling problem [28] and the quadratic assignment problem [29]. Besides its original domain of combinatorial optimization, ACO is also now used to solve continuous optimization problems [30]. Some extensions of ACO algorithms have been proposed in the literature such as ACS [27], Ant System (AS) [31] and MMAS [32]. Recently there have also been a number of studies extending ACO to the field of multi-objective optimization [33]. These algorithms mainly differ with respect to the three following points.

Pheromone update When updating pheromone trails, one has to decide on which of the constructed solutions to lay pheromones. There are usually two strategies to update the pheromone trails. A first strategy is to select the iteration-best or best-so-far solutions to update the pheromone matrices, with respect to each objective. A second strategy is to collect and store the non-dominated solutions in an external set. Only the solutions in the non-dominated set are allowed to update the pheromones.

Definition of pheromone and heuristic information. At each step of the construction of a solution, a candidate is chosen relative to a transition probability which depends on two factors: a pheromone factor and a heuristic factor. There are two approaches to define the pheromone/heuristic information: using one or multiple matrices. When only one matrix is utilized, the pheromone information associated with each objective is combined to reduce the multiple objectives into a single one. If multiple matrices are used, usually each matrix corresponds to one objective. With respect

to the pheromone information, each matrix may contain different values depending on the implementation strategy applied. The same applies to the heuristic information.

Pheromone and heuristic aggregation. Whenever multiple matrices are used, one must use some form of aggregation procedure to aggregate the pheromone/heuristic matrices. There are three common strategies for this: (1) the weighted sum, where matrices are aggregated by a weighted sum; (2) the weighted product, where matrices are aggregated by a weighted product; and (3) random, where at each step a random objective is selected to be optimized. Whenever weights are used for aggregating multiple matrices, two strategies can be applied for setting the weights used at each iteration of the algorithm: (a) dynamically, where each ant may be assigned a different weight from the other ants at each iteration; (b) fixed, where we can assign to all ants the same weight and each objective has the same importance during the entire algorithm run.

2.3. Evolutionary multi-objective optimization

Multi-objective evolutionary algorithms (MOEAs) are stochastic optimization methods, which usually use a population-based approach to find Pareto optimal solutions [34]. The majority of existing MOEAs use the concept of dominance during selection. Therefore, we focus here on the class of dominance-based MOEAs only. All points which are not dominated by any other point are called the non-dominated points. Usually the non-dominated points together constitute a front in the objective space and are often visualized to represent a non-domination front. The points lying on the non domination front, by definition, do not get dominated by any other point in the objective space, hence they are Pareto optimal points (together they make up the Pareto optimal front), and the corresponding variable vectors are called Pareto optimal solutions.

3. Problem statement and formulation

In a cloud environment, we have a pool of server nodes with applications running on them. Suppose that the cluster is fully virtualized and all the applications are running on VMs. The problem of VM placement across a pool of server nodes is related to the multidimensional vector packing problems. Dimensions in the packing problem are resource utilizations. In our work we use two dimensions to characterize a VM and a server node CPU and memory. We do not consider the disk size dimension because we assume that network-attached storage (NAS) is used as main storage across the cluster. If two VMs are running on the same server, the CPU utilization of the server is estimated as the sum of the CPU utilizations of the two VMs. This is the case with memory resources. For example, let (20%, 30%) be a pair of the CPU and memory requests of a VM, and (35%, 40%) be that of another VM. Then, the utilizations of a server accommodating the two VMs are estimated at (55%, 70%), i.e., the sum of the vectors. To prevent CPU and memory usage of a server from reaching 100%, we have to impose an upper bound on resource utilization of a single server with some threshold value. The main idea behind this is that 100% utilization can cause severe performance degradation and VM live migration technology consumes some amount of CPU processing capability on the migrating node.

3.1. Resource wastage modeling

The remaining resources available on each server may vary greatly with different VM placement solutions. To fully utilize multidimensional resources, the following equation is used to calculate the potential cost of wasted resources:

$$Wj = |Lp - Lmj| + \varepsilon(Upj + Umj)$$

where Wj denotes the resource wastage of the j-th server, Up and Um represent the normalized CPU and memory resource usage (i.e., the ratio of used resource to total resource). Lp and Lm represent the normalized remaining CPU and memory resource. ε is a very small positive real number and its value is set to be 0.0001. The key idea behind the above equation is to make effective use of the resources in all dimensions and balance the resources left on each server along different dimensions.

3.2. Power consumption modeling

Recent studies show that the power consumption of servers can be accurately described by a linear relationship between the power consumption and CPU utilization [38]. This linear relationship is also confirmed by our profiling conducted on a Dell server. In order to save energy, servers are turned off when they are idle. Hence, their idle power is not part of the total energy consumption. Finally, we defined the power consumption of the j-th server as a function of the CPU utilization as shown in Eq

$$P\,j = \_(Pbusyj - Pidlej) \times U\,pj + Pidlej\,,\, UCj > 0$$

where Pidle and Pbusy are the average power values when the jth server is idle and fully utilized, respectively. In our simulation experiments, the values have been fixed to 162 and 215 Watt according to the measurement.

3.3. Optimization formulation

Next, we formalize the VM placement optimization problem. Suppose that we are given n VMs (applications) i ∈ I that are to be placed on m servers j ∈ J. For simplicity, we assume that none of the VMs requires more resource than can be provided by a single server. Let Rp be CPU demand of each VM, Tp be the threshold of CPU utilization associated with

each server, Rm be the memory demand of each VM, and Tm be the threshold of memory utilization associated with each server. We use two binary variables xij and yj. The binary variable xij indicates if VM i is assigned to server j and the binary

variable yj indicates whether server j is in use or not. Our objective is to simultaneously minimize the power consumption and the resource wastage. The placement problem can therefore be formulated as: Minimize=

$$\sum_{j=1}^{m}[Yj * (Pbusy - Pidle) * \sum_{i=1}^{n} xij.Rp\ ) + Pidle$$

4. Description of proposed multi objective ant colony System

The algorithm proposed to solve the problems is mainly based on an ACS. A feasible and complete solution of the formulated multi-objective VM placement problem is considered as a permutation of VM assignment.The terms "host" and "server" will be used interchangeably in this paper. An assignment of a VM to a host is called a movement and represented by VM-Host. The pseudocode of the proposed multi-objective ant colony system algorithm (VMPACS) is depicted in Fig. 2. This algorithm works as follows: in an initialization phase, the parameters are initialized and all the pheromone trails are set to τ0. In the iterative part each ant receives all VM requests, introduces a physical server and starts assigning VMs to hosts. This is achieved by the use of a pseudo-random-proportional rule, which describes the desirability for an ant to choose a particular VM as the next one to pack into its current host. This rule is based on the information about the current pheromone concentration on the movement and a heuristic which guides the ants towards choosing the most promising VMs. A local pheromone update is performed once an artificial ant has built a movement. After all ants have constructed their solutions, a global update is performed with each solution of the current Pareto set.

4.1. Definition of the pheromone trail and the heuristic information

Similarly to the general implementation of ACO algorithms, VMPACS starts with a pheromone trails matrix and a heuristic information matrix. The quality of an ACO implementation depends greatly on the definition of the meaning of the pheromone trail [31]. It is crucial to choose a definition which conforms to the feature of the problem. One may consider two different pheromone structures: one that associates a pheromone trail with every movement VM-Host, or one that associates a pheromone trail with every pair of VMs. In this paper, the first way of laying pheromone trails is used, i.e., the pheromone trail τi,j will be defined as the favorability of packing VM i into host j. In the initialization phase, initial pheromone level is calculated by τ0= 1/[n·(P(S0)+ W(S0))], where n is the number of VMs, S0is the solution generated by the FFD heuristic and W(S0) is the resource wastage of the solution S0. P(S0) is the normalized power consumption of the solution S0and its value is calculated according to the following equation:

P(S0) = ∑_(j=1)^m⟦(Pj/⟦Pj⟧^max

Where $_{pj\ max}$ is the peak power consumption of server j.

Apart from pheromone trails, another important factor in an ACO application is the choice of a good heuristic, which will be used in combination with the pheromone information to build solutions. It guides the probabilistic solution construction of ants with problem-specific knowledge. The heuristic information is denoted by ηi,j. This information indicates the desirability of assigning VM i to host j. In order to accurately assess the desirability of each move, the heuristic information is dynamically computed according to the current state of the ant. Since the heuristic information is calculated for all movements in all ants, it may significantly affect the efficiency of the algorithm. To overcome such difficulties it therefore should be computed in an efficient manner. The proposed method to calculate the heuristic information considers the partial contribution of each move to the objective function value. Let PL be a list composed of all the servers. When constructing a solution, every ant starts with the set of all VMs to be placed and the list PL arranged in randomly order. It initially assigns VMs one by one to the first host in the list PL, then assigns to the second host and so on till all VMs are assigned. Therefore, while calculating the value of ηi,j, the permutation of VM assignments from the host 1 to host j is known.

4.2. Pheromone trail update

Another vital component of VMPACS is the update of pheromone trails. The pheromone trail value can either increase, as ants deposit pheromone, or decrease, due to pheromone evaporation. The deposit of new pheromone is based on the fact that the information contained in some good solutions should be indicated by pheromone trails and the movement included in these good solutions will be biased by other ants constructing subsequent solutions. However, pheromone evaporation also implements a useful form of forgetting: it avoids a too rapid convergence of the algorithm toward a suboptimal region, therefore favoring the exploration of new areas of the search space. It is a kind of diversification strategy. In our proposed algorithm, the pheromone updating process includes two steps: a local pheromone update and a global pheromone update. While constructing an assignment of VM i to host j, an ant decreases the pheromone trail level between VM i and host j.

The global updating rule is applied after all ants have finished building a solution. Since all non-dominated or Pareto solutions are considered as optimal or best solutions for a multi-objective optimization problem, we suppose that all nondominated solutions have the same and highest quality and all dominated solutions must be omitted. The global non-

dominated solutions, that form the Pareto set, are stored in an external set. If a solution in the current iteration is not dominated by any other solutions in the current iteration or the external set of non-dominated solutions, this solution is added to the external set and the quantity of pheromone in all movements which constructed it will be increased. Then all solutions dominated by the added one are eliminated. This global updating rule tries to increase the learning.

5. Computation:

Simulation experiments to evaluate the proposed algorithm with respect to performance and scalability. The performance of the proposed ant algorithm is compared to that of a multi-objective grouping genetic algorithm (MGGA) proposed in [16], a single-objective ACO (SACO) algorithm proposed in [24] and a single-objective FFD algorithm proposed in [40]. The programs for the proposed algorithm, MGGA algorithm and FFD heuristic were coded in the Java language and ran on an Intel Pentium Dual-Core processor with 2.50 GHz CPU and 3 GB RAM. The settings for various parameters in VMPACS have a direct effect on the algorithm performance. Appropriate parameter values were determined on the basis of preliminary experiments. The final parameter settings were determined to be NA = 10, M = 100, $\alpha = 0.45$, $\rho l = \rho g = 0.35$, and $q0 = 0.8$. In the case of the MGGA algorithm the population size is 12. The initial population was generated randomly. The crossover rate is 0.7 and the mutation rate is 0.05.The maximum number of generations for each search process is 10. With the above configuration, we randomly generated problem instances. The instances were a demand set of CPU and memory utilizations for 200 VMs. The number of servers was set to the number of VMs in order to support the worst VM placement scenario, in which only one VM is assigned per server. For simplicity, we simulated homogeneous server environments but the proposed approach can be used for the case of heterogeneous servers. After the VM placement algorithm was finished, if there were several non-dominated solutions, a solution belonging to the set of non-dominated solutions was randomly chosen. Every test was repeated with 20 runs for each instance and the average results over 20 independent runs are reported. We introduced the linear correlations of CPU and memory utilizations into the instances and used the method proposed in [40] to generate random sequences of CPU and memory utilizations in the experiments that correlation value are presented.

6. Conclusion

With the increasing prevalence of large scale cloud computing environments, how to efficiently place VMs into available computing servers has become an essential research problem. In this paper, we propose a multi-objective ant colony system algorithm for the virtual machine placement problem. The goal is to efficiently obtain a set of non-dominated solutions that simultaneously minimizes total resource wastage and power consumption. The proposed algorithm is tested with some instances from the literature. Its solution performance is compared to that of an existing multi-objective grouping genetic algorithm. The results demonstrate that our algorithm is competitive. We also compare our algorithm with two single- objective approaches. The comparison shows that VMPACS is superior to those algorithms. Finally the scalability of the proposed algorithm is verified by means of several experiments.

References

[1] Q. Zhang, L. Cheng, R. Boutaba, Cloud computing: state-of-the-art and research challenges, J. Internet Services Appl. 1 (1) (2010).
[2] M. Randles, D. Lamb, E. Odat, A. Taleb-Bendiab, Distributed redundancy and robustness in complex systems, J. Comput. System Sci. 77 (2) (2011).
[3] D. Kusic, J. Kephart, J. Hanson, N. Kandasamy, G. Jiang, Power and performance management of virtualized computing environments via lookahead control, Cluster Computing 12 (1) (2009).
[4] M. Cardosa, M. Korupolu, A. Singh, Shares and utilities based power consolidation in virtualized server environments, in: Proceedings of IFIP/IEEE Integrated Network Management (IM'09), (2009).
[5] L. Grit, D. Irwin, A. Yumerefendi, J. Chase, Virtual machine hosting for networked clusters: Building the foundations for autonomic orchestration, in: Proceedings of the 2nd International Workshop on Virtualization Technology in Distributed Computing, (2006).
[6] W. Vogels, Beyond server consolidation, ACM Queue 6 (1) (2008) .
[7] K. Li, H. Shen, Proxy placement problem for coordinated en-route transcoding proxy caching, Comput. Systems Sci. Engrg. 19 (6) (2004).
[8] K. Li, H. Shen, Optimal proxy placement for coordinated en-route transcoding proxy caching, IEICE Trans. Inform. Syst. 87 (12) (2004).
[9] K. Li, H. Shen, Optimal placement of web proxies for tree networks, in: Proceedings of the IEEE International Conference on e-Technology, e-Commerce and e-Service, 2004.
[10] K. Li, H. Shen, F. Chin, S. Zheng, Optimal methods for coordinated enroute web caching for tree networks, ACM Trans. Internet Technol. (TOIT) 5 (3) (2005) .
[11] K. Li, H. Shen, F. Chin, W. Zhang, Multimedia object placement for transparent data replication, IEEE Trans. Parallel Distrib. Syst. 18 (2) (2007).
[12] S. Chaisiri, B. Lee, D. Niyato, Optimal virtual machine placement across multiple cloud providers, in: Proceedings of the IEEE Asia-Pacific Services Computing Conference, 2009.
[13] M. Bichler, T. Setzer, B. Speitkamp, Capacity planning for virtualized servers, in: Workshop on Information Technologies and Systems (WITS), 2006.
[14] B. Speitkamp, M. Bichler, A mathematical programming approach for server consolidation problems in virtualized data centers, IEEE Trans. Services Comput. (2010) .

[15] H. Mi, H. Wang, G. Yin, Y. Zhou, D. Shi, L. Yuan, Online self-reconfiguration with performance guarantee for energy-efficient large-scale cloud computing data centers, in: Proceedings of the IEEE International Conference on Services Computing, 2010.

[16] J. Xu, J. Fortes, Multi-objective virtual machine placement in virtualized data center environments, in: Proceedings of the IEEE/ACM International Conference on Green Computing and Communications & 2010 IEEE/ACM International Conference on Cyber, Physical and Social Computing, 2010,.

[17] H. Van, F. Tran, J. Menaud, Performance and power management for cloud infrastructures, in: Proceedings of the IEEE 3rd International Conference on Cloud Computing, 2010.

[18] F. Hermenier, X. Lorca, J. Menaud, G. Muller, J. Lawall, Entropy: a consolidation manager for clusters, in: Proceedings of the 2009 ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments, 2009.

[19] J. Békési, G. Galambos, H. Kellerer, A 5/4 linear time bin packing algorithm, J. Comput. System Sci. 60 (1) (2000).

[20] N. Bobroff, A. Kochut, K. Beaty, Dynamic placement of virtual machines for managing sla violations, in: Proceedings of the 10th IEEE Symposium on Integrated Management (IM), 2007, pp. 119–128.

[21] A. Verma, P. Ahuja, A. Neogi, pMapper: power and migration cost aware application placement in virtualized systems, in: Proceedings of the 9th ACM/IFIP/USENIX International Conference on Middleware, 2008.

[22] S. Srikantaiah, A. Kansal, F. Zhao, Energy aware consolidation for cloud computing, in: Proceedings of HotPower'08 Workshop on Power Aware Computing and Systems, 2008.

[23] B. Li, J. Li, J. Huai, T. Wo, Q. Li, L. Zhong, Enacloud: an energy-saving application live placement approach for cloud computing environments, in: Proceedings of the IEEE International Conference on Cloud Computing, 2009.

[24] E. Feller, L. Rilling, C. Morin, Energy-aware ant colony based workload placement in clouds, in: Proceedings of the IEEE/ACM International Conference on Grid Computing (GRID), 2011.

[25] G. Khanna, K. Beaty, G. Kar, A. Kochut, Application performance management in virtualized server environments, in: Proceedings of the 10th IEEE/IFIP Network Operations and Management Symposium (NOMS), 2006.

[26] C. Lin, G. Wu, F. Xia, M. Li, L. Yao, Z. Pei, Energy efficient ant colony algorithms for data aggregation in wireless sensor networks, J. Comput. System Sci. 78 (6) (2012).

[27] M. Dorigo, L. Gambardella, Ant colony system: A cooperative learning approach to the traveling salesman problem, IEEE Trans. Evol. Comput. 1 (1) (1997).

[28] S. Shyu, B. Lin, P. Yin, Application of ant colony optimization for no-wait flowshop scheduling problem to minimize the total completion time, Comput.Indust. Engrg. 47 (2–3) (2004) .

[29] V. Maniezzo, A. Colorni, The ant system applied to the quadratic assignment problem, IEEE Trans. Knowl. Data Engrg. 11 (5) (1999) .

[30] K. Socha, C. Blum, An ant colony optimization algorithm for continuous optimization: application to feed-forward neural network training, Neural Comput. Appl. 16 (3) (2007).

[31] M. Dorigo, V. Maniezzo, A. Colorni, Ant system: optimization by a colony of cooperating agents, IEEE Trans. Syst. Man Cybernet. Part B: Cybernet- ics 26 (1) (1996).

[32] T. Stutzle, H. Hoos, Max–min ant system and local search for the traveling salesman problem, in: Proceedings of the IEEE International Conference on Evolutionary Computation, 1997,.

[33] C. Garcia-Martinez, O. Cordón, F. Herrera, A taxonomy and an empirical analysis of multiple objective ant colony optimization algorithms for the bi-criteria tsp, European J. Oper. Res. 180 (1) (2007).

[34] M. Ikeda, L. Barolli, A. Koyama, A. Durresi, G. De Marco, J. Iwashige, Performance evaluation of an intelligent cac and routing framework for multimedia applications in broadband networks, J. Comput. System Sci. 72 (7) (2006).

[35] J. Branke, K. Deb, K. Miettinen, Multiobjective Optimization: Interactive and Evolutionary Approaches, Springer-Verlag, New York, 2008.

[36] K. Deb, Multi-Objective Optimization Using Evolutionary Algorithms, Wiley, 2001.

[37] K. Deb, Multi-objective genetic algorithms: Problem difficulties and construction of test problems, Evol. Comput. 7 (3) (1999).

[38] X. Fan, W. Weber, L. Barroso, Power provisioning for a warehouse-sized computer, in: Proceedings of the 34th Annual International Symposium on Computer Architecture, 2007.

[39] V. Maniezzo, Exact and approximate nondeterministic tree-search procedures for the quadratic assignment problem, INFORMS J. Comput. 11 (4) (1999).

[40] Y. Ajiro, A. Tanaka, Improving packing algorithms for server consolidation, in: Proceedings of the International Conference for the Computer Measurement Group (CMG), Computer Measurement Group, 2007.

[41] D. Van Veldhuizen, Multiobjective evolutionary algorithms: Classifications, analyses, and new innovations, PhD thesis, Grad. School of Eng. of the Air Force Institute of Technology, Air University, 1999.

[42] J. Schott, Fault tolerant design using single and multicriteria genetic algorithm optimization, Master's thesis, Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, Cambridge, MA, 1995.

